Die SLICOT-Toolboxen für Matlab

The SLICOT Toolboxes for Matlab

Peter Benner, Daniel Kressner, Vasile Sima und Andras Varga

SLICOT ist eine umfangreiche Softwarebibliothek zur numerischen Behandlung von Fragestellungen aus der System- und Regelungstheorie, die mit dem Ziel entwickelt wurde, hohe Leistungsfähigkeit mit Robustheit, Verlässlichkeit, sowie Benutzerfreundlichkeit zu vereinen. Dies wird mittels einer Kombination von Fortran-Kernroutinen und MATLAB- bzw. Scilab-Schnittstellen erreicht. In dieser Übersicht soll der Funktionsumfang der folgenden SLICOT-Toolboxen beschrieben und erläutert werden: (1) Grundaufgaben der System- und Regelungstheorie, (2) Systemidentifizierung, (3) Modell- und Reglerreduktion. Der Einsatz der Toolboxen in der Praxis wird durch verschiedene Beispiele veranschaulicht.

SLICOT is a comprehensive numerical software package for control systems analysis and design. While based on highly performant Fortran routines, MATLAB and Scilab interfaces provide convenient alternative access for users. In this survey, we summarize the functionality contained in the three SLICOT toolboxes for (1) basic tasks in systems and control, (2) system identification, and (3) model and controller reduction. Several examples illustrate the use of these toolboxes for addressing frequent computational tasks.

Schlagwörter: Numerische Verfahren, Software, MATLAB, Matrixgleichungen, Systemidentifizierung, Modellreduktion.

Keywords: Numerical methods, software, MATLAB, matrix equations, system identification, model reduction.

1 Einleitung

Die ständig wachsende Komplexität von Regelungssystemen stellt vielfältige Herausfordungen an die Effizienz der eingesetzten numerischen Verfahren. Idealerweise sollte eine Softwarebibliothek in diesem Bereich hohe Leistungsfähigkeit mit Robustheit, Verlässlichkeit, sowie Benutzerfreundlichkeit vereinen. SLICOT[1] [9] wurde mit dem Ziel entwickelt, diesen Anforderungen gerecht zu werden. Diese Arbeit fasst Teile der von SLICOT bereitgestellten Funktionalität zusammen und veranschaulicht deren Einsatz bei der Analyse und Synthese von Regelungssystemen.

Alle rechenintensiven Grundroutinen von SLICOT sind aus Effizienz-, Robustheits-, und Portabilitätserwägungen in der prozeduralen Programmiersprache Fortran 77 umgesetzt. Um dennoch eine einfache Benutzung gewährleisten zu können, werden, basierend auf diesen Grundroutinen, drei MATLAB[2]-Toolboxen bereitgestellt:

(1) Grundaufgaben der System- und Regelungstheorie,
(2) Systemidentifizierung,
(3) Modellreduktion.

Die in diesen Toolboxen enthaltenen MATLAB-Funktionen (M-Funktionen) rufen mittels sogenannter MEX-Schnittstellen die Fortran-Routinen von SLICOT auf.[3] Ziel dieser Arbeit ist es, einen Überblick über die

[1] Die SLICOT-Bibliothek (Subroutine Library in Systems and Control Theory) wurde innerhalb des durch die Europäische Gemeinschaft (BRITE-EURAM III RTD Thematic Networks Programme) geförderten Netzwerkes NICONET (Numerics in Control Network) entwickelt, siehe `http://www.icm.tu-bs.de/NICONET`. Akademischen und nichtkommerziellen Anwendern wird SLICOT unter `http://www.slicot.org` zur freien Nutzung überlassen.

[2] MATLAB ist eingetragenes Warenzeichen von The MathWorks.

[3] Es ist sogar möglich, wenngleich mühsamer, die MEX-Schnittstellen direkt aus MATLAB heraus aufzurufen. Solche direkten Aufrufe unter Umgehung der komfortableren M-Funktionen resultieren mitunter in einer höheren Flexi-

Handhabung dieser Toolboxen zu geben und deren Einsatz an verschiedenen Beispielen zu veranschaulichen.

Der Rest dieser Arbeit ist wie folgt gegliedert. Abschnitt 2 beschreibt, teilweise recht detailliert, die Handhabung von Toolbox (1) zur Lösung grundlegender Fragestellungen, wie etwa der Steuerbarkeit oder der Beobachtbarkeit eines linearen zeitunabhängigen Systems. Die Abschnitte 3 und 4 geben eine breitere und weniger detailliertere Zusammenfassung der beiden anderen Toolboxen, wobei insbesondere deren Vorteile bei der Lösung anspruchsvollerer Probleme betont werden sollen.

2 Grundaufgaben der System- und Regelungstheorie

Vorwiegend bezieht sich die in SLICOT enthaltene Funktionalität auf *lineare zeitunabhängige (LTI) Systeme in Zustandsraumdarstellung*. Im kontinuierlichen Fall haben diese die Form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

und im diskreten Fall

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k. \end{aligned} \tag{2}$$

Dabei wird die $n \times n$ Matrix A als *Systemmatrix* bezeichnet, die $n \times m$ Matrix B als *Eingangsmatrix*, die $p \times n$ Matrix C als *Ausgangsmatrix*, und die $p \times m$ Matrix D als *Durchgangsmatrix*. Im folgenden werden nur Matrizen mit reellen Einträgen betrachtet.

Die für MATLAB erhältliche Control System Toolbox [26, 3] basiert auf einem LTI-Objekt, mit dem sich lineare Zustandsraummodelle komfortabel speichern und manipulieren lassen. Zum Beispiel erzeugt der Befehl `sys = ss(A,B,C,D)` ein kontinuierliches LTI-Objekt (1) mit den Matrizen A, B, C, D. SLICOT unterstützt solche LTI-Objekte. Es soll aber betont werden, dass auf die gesamte Funktionalität von SLICOT auch ohne LTI-Objekte (und insbesondere ohne Verfügbarkeit der Control System Toolbox) zugegriffen werden kann.

Übertragungsfunktionen bieten eine alternative Darstellungsmöglichkeit von LTI-Systemen und können aus (1) bzw. (2) mittels der Laplace- bzw. z-Transformation gewonnen werden:

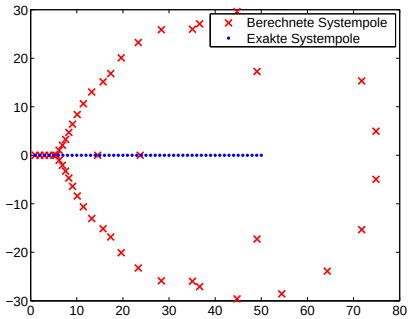
$$G(\lambda) = C(\lambda I - A)^{-1}B + D, \tag{3}$$

wobei λ die in der Transformation auftretende Variable bezeichnet. Im praktisch relevanten Fall $m, p \ll n$ ist diese Darstellung sehr kompakt; sie benötigt lediglich $O(nmp)$ Parameter für die Koeffizienten der Nenner und Zähler der matrixwertigen rationalen Funktion $G(\lambda)$. Im Gegensatz dazu führt die Zustandsraumdarstellung, allein schon wegen der Systemmatrix A, zu $O(n^2)$ Einträgen. Nichtsdestotrotz greift SLICOT in Berechnungsroutinen niemals auf die Übertragungsfunktionsdarstellung zurück, da die sich daraus ergebenden schwerwiegenden numerischen Probleme lediglich die Behandlung sehr kleiner Systeme erlauben würden.

Beispiel 1
Der folgende MATLAB*-Quelltext konvertiert die Zustandsraumdarstellung eines Systems mit diagonaler Systemmatrix in eine Übertragungsfunktion:*

```
A = diag(1:50); B = ones(50,1); C = ones(1,50);
sys = ss(A,B,C,0); tra = tf(sys);
```

Der Befehl `eig(tra)` *berechnet die Pole dieser Übertragungsfunktion (in der folgenden Darstellung mit roten Kreuzen gekennzeichnet).*



Es stellt sich heraus, dass die bei der Konvertierung in Übertragungsfunktionsdarstellung auftretenden Rundungsfehler die exakten Pole $1, \ldots, 50$ *(mit blauen Punkten gekennzeichnet) weitestgehend zerstören.*

Die ebenfalls von SLICOT unterstützten aber im weiteren nicht näher behandelten *Deskriptorsysteme* [27, 36] erhält man, wenn die Zustandsgleichungen in (1) und (2) durch die allgemeineren Gleichungen $E\dot{x}(t) = Ax(t) + Bu(t)$ beziehungsweise $Ex_{k+1} = Ax_k + Bu_k$ mit $E \in \mathbb{R}^{n\times n}$ ersetzt werden. Das entsprechende LTI-Objekt wird mit `sys = dss(A,B,C,D,E)` erzeugt.

2.1 System-Analyse

Dieser Abschnitt stellt einige SLICOT-Funktionen zur Analyse systeminherenter Eigenschaften vor.

2.1.1 Pole und Nullstellen

Die Funktion `polzer` berechnet die Pole und Nullstellen, sowie den Normalrang eines (Deskriptor-)Systems.

Beispiel 2 *Wir betrachten ein lineares System mit Polen* $-1/2, 1$*, Nullstelle* $-1/2$ *und Normalrang* 1*:*

```
A = [4 3;-9/2 -7/2]; B = [ 1; -1 ]; C = [3 2];
```

bilität bei der Übergabe von Datenstrukturen, was wiederum höhere Effizienz nach sich ziehen kann. In SLICOT sind vorkompilierte MEX-Schnittstellen für die Betriebssysteme WINDOWS, Sun Solaris und Linux verfügbar.

```
D = 0; sys = ss(A,B,C,D);
```
Der Befehl `[p,z,r] = polzer(sys)` *gibt das folgende Ergebnis zurück:*

```
p =              z =              r =
    1.0000           -0.5000          1
   -0.5000
```

Optional berechnet die Funktion `polzer` auch die Kronecker-Struktur [28] des zu einem Deskriptorsystem gehörigen Systembüschels $\begin{bmatrix} A-\lambda E & B \\ C & D \end{bmatrix}$.

2.1.2 Steuerbarkeit und Beobachtbarkeit

Mit der folgenden, auf orthogonalen Transformationen basierenden Variante der *Regelungsnormalform* (auch: orthogonale Kalman-Zerlegung) lässt sich numerisch zuverlässig die Steuerbarkeit eines Systems überprüfen oder gegebenenfalls nicht steuerbare Pole identifizieren. Dabei wird eine orthogonale Matrix U berechnet, so dass

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ 0 & A_{22} & 0 \\ \hline C_1 & C_2 & D \end{array}\right], \tag{4}$$

wobei das Untersystem $\begin{bmatrix} A_{11} & B_1 \\ C_1 & D \end{bmatrix}$ steuerbar ist und A_{22} alle nicht steuerbaren Pole enthält. Weiterhin befinden sich A_{11} und A_{22} in Treppennormalform[4]; nähere Details dazu finden sich z.B. in [17]. Offensichtlich ist das ursprüngliche System genau dann steuerbar, wenn der Block A_{22} nicht vorhanden ist. Die Zerlegung (4) wird von der SLICOT-Funktion `[syscf,Nc{,U,s}] = slconf(sys{,tol})`[5] berechnet. Dabei enthält das LTI-Objekt `syscf` das transformierte System (4) und die natürliche Zahl `Nc` entspricht der Ordnung des steuerbaren Untersystems. Die optionalen Ausgabeargumente `U` und `s` enthalten die orthogonale Transformationsmatrix bzw. die Blockgrößen in der Treppennormalform. Der optionale Parameter `tol` bestimmt die Toleranz mit der während der Berechnung Rangentscheidungen getroffen werden (Voreinstellung `tol` $= n^2 \times$ `eps`).

Beispiel 3 `[syscf,Nc] = slconf(sys)` *mit dem LTI-Objekt* `sys` *aus Beispiel 2 liefert:*

```
a =                    b =
        x1     x2              u1
   x1    1   -7.5       x1  -1.414
   x2    0   -0.5       x2       0

c =                          d =
           x1        x2             u1
   y1  -0.7071     3.536        y1   0

Nc =
     1
```

Der Pol -0.5 *ist also nicht steuerbar.*

Die orthogonale Variante der *Beobachternormalform* ist eine Zerlegung der Form

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{cc|c} A_{11} & 0 & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & 0 & D \end{array}\right], \tag{5}$$

wobei U wieder orthogonal ist und das Untersystem $\begin{bmatrix} A_{11} & B_1 \\ C_1 & D \end{bmatrix}$ beobachtbar ist. Die Matrix A_{22} enthält alle nicht beobachtbaren Pole. Die Syntax der entsprechenden SLICOT-Funktion `slobsf` ist analog zu der von `slconf`.

Durch Kombination von Regelungs- und Beobachternormalform erhält man die Zerlegung

$$\left[\begin{array}{c|c} U^TAU & U^TB \\ \hline CU & D \end{array}\right] = \left[\begin{array}{ccc|c} A_{11} & A_{12} & A_{13} & B_1 \\ 0 & A_{22} & A_{23} & B_2 \\ 0 & 0 & A_{33} & 0 \\ \hline 0 & C_2 & C_3 & D \end{array}\right]. \tag{6}$$

Das steuerbare und beobachtbare Untersystem $\begin{bmatrix} A_{22} & B_2 \\ C_2 & D \end{bmatrix}$ ist eine minimale Realisierung der Übertragungsfunktion (3) und kann mit `slminr` berechnet werden.

2.1.3 Systemnormen

Durch Aufruf der SLICOT-Funktion `slh2norm(sys,2)` wird die L_2-Norm eines kontinuierlichen oder diskreten linearen Systems, welches keine Pole auf der imaginären Achse bzw. auf dem Einheitskreis besitzt, berechnet. Im kontinuierlichen Fall ist die L_2-Norm definiert durch

$$\|G\|_2 := \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \|G(\imath\omega)\|_F^2 \, \mathrm{d}\omega},$$

wobei G die Übertragungsfunktion (3) des Systems und $\|\cdot\|_F$ die Frobenius-Norm bezeichnen. Für asymptotisch stabile Systeme wird die L_2-Norm zur in den Anwendungen nützlicheren H_2-Norm. Zur Berechnung der H_2-Norm reicht der Aufruf `slh2norm(sys)`, wobei dann die asymptotische Stabilität ebenso vorausgesetzt wird wie $D = 0$. Die Berechnung erfolgt in diesem Fall über die Darstellung (siehe [19, Abschnitt 3.3.3])

$$\|G\|_2 := \sqrt{\mathrm{Spur}(B^TW_oB)},$$

wobei W_o die Beobachtbarkeits-Gramsche des Systems ist. Diese lässt sich mit Hilfe der zweiten Lyapunovgleichung in (12) (siehe Abschnitt 4) berechnen, wozu dann die Methoden aus Abschnitt 2.2 verwendet werden können.

[4] Für Systeme mit nur einem Eingang entspricht die Treppennormalform der oberen Hessenbergform. Die Normalform entspricht dann der System-Hessenberg-Form.

[5] Hier und im folgenden sind optionale Ein- und Ausgabeparameter in geschweifte Klammern gefasst.

Für kontinuierliche lineare Systeme ohne Pole auf der imaginären Achse berechnet die Funktion `slinorm(sys)` die L_∞-Norm

$$\|G\|_\infty = \sup_{\omega\in\mathbb{R}} \sigma_{\max}\big(G(\imath\omega)\big)$$

(siehe z.B. [19, Abschnitt 3.2.2]), wobei $\sigma_{\max}$ den maximalen Singulärwert einer Matrix bezeichnet. Für asymptotisch stabile Systeme berechnet `slinorm(sys)` die H_∞-Norm. Die L_∞-Norm instabiler Systeme wird in der Praxis zwar seltener benötigt als die H_∞-Norm stabiler Systeme, kann aber z.B. bei der Modellreduktion als Maß für die Approximationsgüte der Übertragungsfunktion dienen. Die in [13, 15] vorgeschlagenen Algorithmen funktionieren für stabile und instabile Systeme gleich, daher liegt es nahe, die volle Funktionalität (wie auch beim MATLAB-Befehl `norm(sys,inf)`) zur Verfügung zu stellen. Mit einem zusätzlichen Ausgabeargument `[ninf,fpeak] = slinorm(sys)` kann auch die Frequenz `fpeak` bestimmt werden, an der $\|G(\imath\omega)\|_2$ den Wert $\|G\|_\infty$ annimmt.

Für diskrete Systeme berechnen `slh2norm` und `slinorm` die Normen

$$\|G\|_2 = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi} \|G(e^{\imath\theta})\|_F^2 \,\mathrm{d}\theta},$$
$$\|G\|_\infty = \sup_{\theta\in[-\pi,\pi]} \sigma_{\max}\big(G(e^{\imath\theta})\big).$$

Die Funktion `slhknorm` berechnet die Hankel-Norm eines asymptotisch stabilen Systems. Bei einem instabilen System berechnet `slhknorm` die Hankel-Norm des stabilen Untersystems.

2.2 Lineare Matrixgleichungen

Lyapunov-Gleichungen der Form

$$A^TX + XA = W, \tag{7}$$

mit einer $n\times n$ Matrix A und einer symmetrischen $n\times n$ Matrix W treten in verschiedenen Anwendungen der Regelungstheorie auf, siehe z.B. Abschnitt 4. Hat A keine rein imaginären Eigenwerte, so ist die Existenz, Eindeutigkeit und Symmetrie der Lösung X von (7) gesichert und X kann mit dem Befehl `X = sllyap(A,W)` berechnet werden. Ist darüber hinaus die Koeffizientenmatrix A asymptotisch stabil und die rechte Seite W negativ semidefinit, so ist X positiv semidefinit und besitzt also eine Cholesky-ähnliche Faktorisierung der Form $X = R^TR$. Ist die rechte Seite ebenfalls in faktorisierter Form $W = -C^TC$ gegeben, so erlaubt der Funktionsaufruf `R = slstly(A,C)` die direkte Berechnung des Lösungsfaktors R, unter Umgehung der Berechnung von X [20].

Beide SLICOT-Funktionen, `sllyap` und `slstly`, stellen weitere optionale Eingabeparameter zur Verfügung, mit denen die zu (7) duale Gleichung $AX+XA^T = W$ gelöst und Dreiecksstrukturen in A ausgenutzt werden können. So ist es beispielsweise möglich, die beiden Lyapunov-Gleichungen

$$A(R_cR_c^T) + (R_cR_c^T)A = -BB^T,$$
$$A^T(R_o^TR_o) + (R_o^TR_o)A = -C^TC,$$

gleichzeitig, mit nur einer Schur-Zerlegung von A, zu lösen:

```
[Q,T] = schur(A);
Rc = Q * slstly(T,Q'*B,1,1);
Ro = slstly(T,C*Q,1,0) * Q';
```

In einem weiteren Ausgabeparameter, `[X,sep] = sllyap(A,C{,struct,trans})`, kann die Separation

$$\mathrm{sep}(A,-A^T) = \min_{\|X\|_F=1} \|A^TX + XA\|,$$

zurückgegeben werden. Das Reziproke der Separation einer Lyapunov-Gleichung bietet ein Maß für die zu erwartende Genauigkeit der berechneten Lösung [21].

Tabelle 1 fasst alle in SLICOT enthaltenen Funktionen zur Lösung von linearen Matrixgleichungen zusammen. Wie oben illustriert kann mit dem optionalen Eingabeparameter `struct` angegeben werden, dass sich Koeffizienten bereits in reduzierter Hessenberg- bzw. Schur-Form befinden. Ein ausführlicher Vergleich der Funktionen mit anderer Software findet sich in [12].

2.3 Quadratische Matrixgleichungen

Im kontinuierlichen Fall hat die *algebraische Matrix-Riccati-Gleichung* die Form

$$0 = Q + A^TX + XA - (L + XB)R^{-1}(L + XB)^T \tag{8}$$

mit $Q\in\mathbb{R}^{n\times n}$ symmetrisch, $R\in\mathbb{R}^{m\times m}$ symmetrisch und invertierbar, und $A\in\mathbb{R}^{n\times n}$, $B\in\mathbb{R}^{n\times m}$, $L\in\mathbb{R}^{n\times m}$. Bereits relativ schwache Annahmen (siehe z.B. [24]) genügen, um die Existenz, Eindeutigkeit und Symmetrie der für die System- und Regelungstheorie relevanten *stabilisierenden* Lösung X von (8) zu sichern. Über den Befehl `X = slcares(A,Q,R,B,L)` wird X mittels der Schur-Methode nach Laub [25] berechnet. Dieser Algorithmus gewinnt $X = X_2X_1^{-1}$ aus einer Basis $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ des stabilen invarianten Unterraums der zu (8) gehörigen Hamiltonischen Matrix H. Die entsprechenden stabilen Eigenwerte von H können in einem weiteren Ausgabeparameter zurückgegeben werden: `[X,ev] = slcares(A,Q,R,B,L)`. Der Aufruf `[X{,sep}] = slcares(A,Q,G)` behandelt den folgenden wichtigen Spezialfall von (8):

$$0 = Q + A^TX + XA - XGX.$$

Die SLICOT-Funktion `carecond` berechnet die Konditionszahl von (8).

Tabelle 1: Löser für lineare Matrixgleichungen in SLICOT.

Lyapunov-Gleichung	$A^TX + XA = W$	`[X{,sep}] = sllyap(A,W{,struct})`
	$AX + XA^T = W$	`[X{,sep}] = sllyap(A,W,struct,1)`
	$A^TR_o^TR_o + R_o^TR_oA = -C^TC$	`Ro = slstly(A,C{,struct})`
	$AR_cR_c^T + R_cR_c^TA^T = -BB^T$	`Rc = slstly(A,B,struct,1)`
Stein-Gleichung	$A^TXA - X = W$	`[X{,sep}] = slstei(A,W{,struct})`
	$AXA^T - X = W$	`[X{,sep}] = slstei(A,W,struct,1)`
	$A^TR_o^TR_oA - R_o^TR_o = -C^TC$	`Ro = slstst(A,C{,struct})`
	$AR_cR_c^TA^T - R_cR_c^T = -BB^T$	`Rc = slstst(A,B,struct,1)`
Sylvester-Gleichung	$AX + XB = C$	`X = slsylv(A,B,C{,struct})`
	$A^TX + XB^T = C$	`X = slsylv(A,B,C,struct,[1 1])`
	$AXB + X = C$	`X = sldsyl(A,B,C{,struct})`
	$A^TXB^T + X = C$	`X = sldsyl(A,B,C,struct,[1 1])`
Verallgemeinerte Lyapunov-Gleichung	$A^TXE + E^TXA = W$	`[X{,sep}] = slgely(A,E,W{,struct})`
	$AXE^T + EXA^T = W$	`[X{,sep}] = slgely(A,E,W,struct,1)`
	$A^TR_o^TR_oE + E^TR_o^TR_oA = -C^TC$	`Ro = slgsly(A,E,C{,struct})`
	$AR_cR_c^TE^T + ER_cR_c^TA^T = -BB^T$	`Rc = slgsly(A,E,B,struct,1)`
Verallgemeinerte Stein-Gleichung	$A^TXA - E^TXE = W$	`[X{,sep}] = slgest(A,E,W{,struct})`
	$AXA^T - EXE^T = W$	`[X{,sep}] = slgest(A,E,W,struct,1)`
	$A^TR_o^TR_oA - E^TR_o^TR_oE = -C^TC$	`Ro = slgsst(A,E,C{,struct})`
	$AR_cR_c^TA^T - ER_cR_c^TE^T = -BB^T$	`Rc = slgsst(A,E,B,struct,1)`

Ganz analog zu `slcares` kann die SLICOT-Funktion `sldares` zur Lösung des diskreten Falls der algebraischen Matrix-Riccati-Gleichung

$$X = A^TXA - R(X) + Q \tag{9}$$

mit $R(X) = (L + A^TXB)(R + B^TXB)^{-1}(L + A^TXB)^T$ bzw. dem Spezialfall

$$X = Q + A^TX(I + GX)^{-1}A.$$

eingesetzt werden.

Die Funktionen `slgcare` und `slgdare` behandeln die für Deskriptorsysteme relevanten Varianten von (8) bzw. (9). Detailierte Informationen zur Lösung von Riccati-Gleichungen mit SLICOT können der Arbeit [11] entnommen werden.

2.4 Strukturierte Matrixzerlegungen

SLICOT stellt schnelle Löser für lineare Gleichungssysteme und Ausgleichsprobleme mit strukturierten Matrizen bereit. Der Schwerpunkt liegt dabei auf Block-Toeplitz-Matrizen der Form

$$T = \begin{bmatrix} T_0 & T_1 & \cdots & T_l \\ T_{-1} & T_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_1 \\ T_{-k} & \cdots & T_{-1} & T_0 \end{bmatrix}, \tag{10}$$

wobei alle Blöcke $T_{-k}, \ldots, T_{-1}, T_0, T_1, \ldots, T_l$ gleich groß sind.

Sei T_{col} die erste Blockspalte einer quadratischen und symmetrisch positiv definiten Block-Toeplitz-Matrix T. Dann berechnet die Funktion `R = fstchol(Tcol)` die Cholesky-Zerlegung $T = R^TR$ mit oberer Dreiecksmatrix R. Mit zusätzlichen Parametern, `[R,X] = fstchol(Tcol,B)`, wird die Lösung des linearen Gleichungssystems $TX = B$ berechnet. Die Funktion `X = fstsol(Tcol,B)` löst $TX = B$ direkt, ohne die aufwändige Speicherung des Faktors R.

Für eine allgemeine Block-Toeplitz-Matrix T, mit erster Blockspalte T_{col} und erster Blockzeile T_{row}, gibt `[Q,R] = fstqr(Tcol,Trow)` eine QR-Zerlegung $T = QR$, mit Q orthogonal und R in oberer Dreiecksform, zurück. Unter der Annahme dass T vollen Spaltenrang hat, bestimmt `[Q,R,X{,Y}] = fstqr(Tcol,Trow,B{,C})` außerdem die Lösungen X und Y der linearen Ausgleichsprobleme $\min_X \|TX - B\|_F$ und $\min_{T^TY=C} \|Y\|_F$. Mit dem Aufruf `[R{,X,Y}] = fstqr(Tcol,Trow{,B,C})` wird die Berechnung von Q ausgelassen und – sollte T selbst bandbeschränkt sein – ein bandbeschränkter, dünnbesetzter Cholesky-Faktor R zurückgegeben. Die SLICOT-Funktion `[X,Y] = fstlsq(Tcol,Trow,B,C)` löst beide Ausgleichsprobleme direkt, ohne die aufwändige Speicherung von Q und R.

Die oben beschriebenen Funktionen basieren auf Varianten des verallgemeinerten Schur-Algorithmus [22], dessen Komplexität lediglich quadratisch (anstatt kubisch) in den Matrixdimensionen wächst. Für weitere Details verweisen wir auf [23]. Die SLICOT-Funktion `fstmul(Tcol,Trow,B)` berechnet ein Matrix-Produkt der Form TB mittels schneller Hartley-Transformationen [14].

2.5 Benchmark-Sammlungen

SLICOT stellt 5 Sammlungen von verschiedenen akademischen und praktisch relevanten Benchmark-Beispielen

zur Verfügung, mit denen die Genauigkeit und Performance von Algorithmen überprüft werden können. Darüber hinaus erlauben diese Benchmarks Einblicke in das Verhalten eines Verfahrens in extremen Situationen, wenn z.B. die Grenzen der Maschinengenauigkeit erreicht werden.

Das MATLAB-Skript `aredata` stellte die Daten von algebraischen Riccati-Gleichungen bereit: 21 Beispiele für den kontinuierlichen Fall (8) und 19 Beispiele für den diskreten Fall (9). Einige dieser Beispiele hängen von Parametern ab, mit denen die Dimension oder gewisse Eigenschaften des Problems verändert werden können, siehe [1, 2]. Die nah verwandten Funktionen `ctdsx` und `dtdsx` enthalten Benchmark-Beispiele für kontinuierliche bzw. diskrete LTI-Systeme. Analog enthalten die Funktionen `ctlex` und `dtlex` Benchmark-Beispiele für (verallgemeinerte) Lyapunov-Gleichungen.

3 Systemidentifizierung

Die SLICOT System Identification Toolbox stellt MATLAB- und Fortran-Routinen für diskrete zeitinvariante lineare Systeme (2) und nichtlineare Systeme vom Wiener-Typ [40] zur Verfügung. Zur Identifizierung von linearen Systemen werden die weit verbreiteten Unterraum-basierten Zugänge MOESP [39] (Multivariable Output-Error state SPace identification) und N4SID [34] (Numerical algorithms for Subspace State Space System IDentification) verwendet. Bei den nichtlinearen Systemen kommen diese Zugänge ebenfalls zur Initialisierung der Parameter des linearen dynamischen Anteils zum Einsatz. Der nichtlineare Beobachteranteil hingegen wird als einschichtiges neuronales Netz modelliert und mit Hilfe des Levenberg-Marquardt-Algorithmus geschätzt.

Der Funktionsumfang der Toolbox enthält:

- Identifizierung von diskreten LTI-Systemen;
- Identifizierung der Zustands- und Beobachter-Kovarianzmatrizen von diskreten LTI-Systemen;
- Schätzung der zugehörigen Kalman-Gain-Matrix;
- Schätzung des Anfangszustands;
- Konvertierung von Zustands- in Beobachternormalform, und umgekehrt;
- Identifizierung von diskreten Wiener-Systemen;
- Berechnung der Beobachterantwort von Wiener-Systemen.

Einige besondere Merkmale der Toolbox:

- Geschwindigkeitsvorteile gegenüber Standardsoftware;
- verschiedene Techniken zur Datenkomprimierung, teilweise unter Ausnutzung von Block-Hankel-Strukturen;
- Batch-Verarbeitung von mehreren, möglicherweise zusammenhängenden Datensätzen;
- spezialisierte strukturausnutzende Varianten des Levenberg-Marquardt-Algorithmus;
- optionale Überwachung der Genauigkeit von Zwischenresultaten mittels der zugehörigen Konditionszahlen.

Insgesamt enthält die Toolbox 14 MATLAB-Dateien, wovon die 4 Hauptfunktionen `slmoesp`, `sln4sid`, `slmoen4`, und `slmoesm`, die vorgestellten Methoden zur Systemidentifizierung durchführen.

3.1 Performance-Resultate

Die Genauigkeit und Effizienz der Toolbox wurde ausführlich mit den Datensätzen von DaISy[6] getestet. Eine detaillierte Übersicht dieser Tests zu 25 verschiedenen Anwendungen findet sich in [33, 32]. Abbildung 1 gibt davon einen repräsentativen Auszug: ein Laufzeitvergleich der SLICOT-Funktion `slmoen4` mit der MATLAB-Funktion `n4sid`. Die Graphik zeigt die gemessene Laufzeit von MATLAB geteilt durch die gemessene Laufzeit von SLICOT. Für einige Anwendungen zeigt sich hierbei ein enormer Geschwindigkeitsvorteil um bis zu zwei Grössenordnungen, der sich teilweise auf den Einsatz von schnellen strukturausnutzenden QR-Zerlegungen in SLICOT zurückführen lässt.

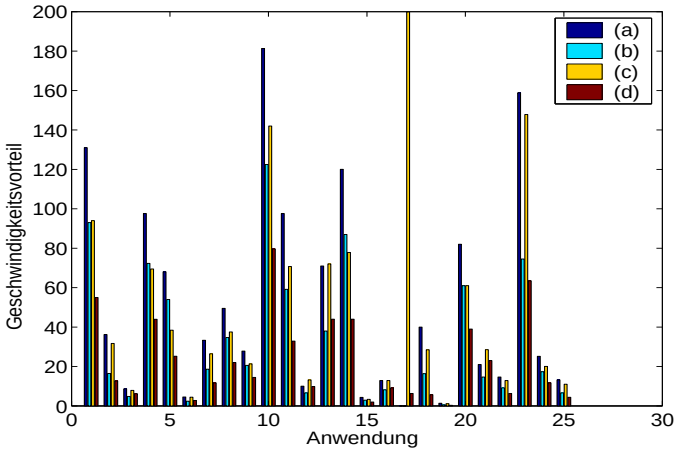


Bild 1: Die SLICOT-Funktion `slmoen4` im Vergleich zur MATLAB-Funktion `n4sid`. Von den Standardeinstellungen abweichende Einstellugen: `order` $= n$, `'N4Weight'` = `'MOESP'`, and: (a) `'Cov'` := `'CovarianceMatrix'` = `[ ]`, `'N4H'` := `'N4Horizon'` = `'Auto'`; (b) `'Cov'` = `'None'`, `'N4H'` = `'Auto'`; (c) `'Cov'` = `[ ]`, `'N4H'` = $[\,s\,s\,s\,]$; (d) `'Cov'` = `'None'`, `'N4H'` = $[\,s\,s\,s\,]$, siehe auch [33].

Performance-Resultate für die Identifizierung von Wiener-Systemen finden sich in [31]. Für detaillierte Informationen zu den in der Toolbox eingesetzten Algorithmen verweisen wir auf die unter `http://www.slicot.org/` verfügbaren SLICOT Working Notes 1998-6, 1999-3, 1999-19, 2000-4, und 2002-6.

[6] DaISy (Database for the Identification of Systems) ist eine frei verfügbare Sammlung von anwendungsnahen Eingangs-Ausgangs-Datensätzen, siehe `http://www.esat.kuleuven.ac.be/sista/daisy`.

4 Modell- und Reglerreduktion

Modell- oder Ordnungsreduktion wird immer mehr zu einem wichtigen Hilfsmittel bei der Simulation, Steuerung, Regelung und Optimierung komplexer naturwissenschaftlicher, technologischer oder ökonomischer Prozesse. SLICOT stellt eine große Vielfalt von Modellreduktionstechniken für LTI-Systeme der Form (1) oder (2) zur Verfügung.

Insbesondere im Bereich des Reglerentwurfs ist Ordnungsreduktion oft unverzichtbar, da moderne und robuste Regler wie z.B. LQG-, H_2- oder H_∞-Regler selbst LTI Systeme der Form (1) oder (2) mit einer Ordnung $N \geq n$ sind. Da in der technologischen Umsetzung in Hardware aufgrund vielfältiger Restriktionen an Echtzeitverhalten und Zuverlässigkeit jedoch nur sehr niedrigdimensionale Regler realisiert werden können, ist schon bei moderaten Zustandsraumdimensionen eine Reduktion nötig. Die Reduktion des Modells, d.h. die Reduktion von n, kann durch Modellreduktion erreicht werden, während die Ordnung N des Reglers direkt mit Methoden zur *Reglerreduktion* erfolgen kann, siehe z.B. [29, 35, 37, 38]. Im Gegensatz zur MATLAB Control System Toolbox bzw. Robust Control Toolbox, enthält SLICOT eine ganze Reihe verschiedener Reglerreduktionsmethoden.

Neben dem Reglerentwurf ist Modellreduktion auch ein wichtiges Werkzeug in Anwendungsbereichen wie Schaltkreisentwurf, Strukturdynamik, numerischer Strömungsmechanik, sowie beim Design von mikro- und nanoelektromechanischen Systemen (MEMS/NEMS). In all diesen Bereichen können die SLICOT Modellreduktiones-Funktionen natürlich ebenfalls eingesetzt werden.

Um die Beschreibung zu vereinfachen, konzentrieren wir uns im Folgenden auf zeitkontinuierliche Systeme, bemerken aber, dass die meisten SLICOT Modell- und Reglerreduktions-Funktionen auch auf zeitdiskrete Systeme angewendet werden können. Dabei wird in manchen Fällen, wenn keine eigene Implementierung für den diskreten Fall existiert (oder die Methode keine algorithmische Entsprechung im diskreten Fall besitzt), zunächst eine bilineare Transformation ins Zeitkontinuierliche durchgeführt, dann das enstandene System der Form (1) reduziert, woraus dann durch eine inverse bilineare Transformation ein reduziertes zeitdiskretes System gewonnen wird.

Unter Modellreduktion versteht man die Aufgabe ein LTI-System

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + Du(t), \end{aligned} \tag{11}$$

der Ordnung r, mit $r \ll n$, zu finden, so dass die zugehörige Übertragungsfunktion $\hat{G}(s) = \hat{C}(sI_r - \hat{A})^{-1}\hat{B} + D$ die originale Übertragungsfunktion $G(s)$ "gut" approximiert. Dies wird motiviert durch die Beziehung $\|y - \hat{y}\|_2 \leq \|G - \hat{G}\|_\infty \|u\|_2$, wobei $\|\cdot\|_2$ die L_2-Norm und $\|\cdot\|_\infty$ die in Abschnitt 2.1.3 diskutierte H_∞-Norm bezeichnet. Man beachte, dass bei der Verwendung der H_∞-Norm die Stabilität des Systems vorausgesetzt werden muss. Die Modell- und Reglerreduktion ist jedoch ebenfalls für instabile Systeme möglich. In SLICOT wird dies entweder durch additive Zerlegung der Übertragungsfunktion in den stabilen und instabilen Anteil oder durch teilerfremde Faktorisierung (mit stabilen Faktoren) und anschliessender Anwendung der Methoden für stabile LTI-Systeme auf die resultierenden stabilen Anteile der Übertragungsfunktion realisiert, siehe z.B. [37].

Methoden, die auf die Minimierung von $\|G - \hat{G}\|$ abzielen, werden *Absolutfehler-Methoden* genannt, wohingegen *Relativfehler-Methoden* versuchen, $\|\Delta_r\|$ zu minimieren, wobei Δ_r implizit durch $G - \hat{G} = \Delta_r G$ definiert ist. Allerdings ist für die H_∞-Norm die effiziente Berechnung solcher Bestapproximationen nach wie vor ein ungelöstes Problem. Deswegen kommen üblicherweise die Methode des balancierten Abschneidens oder verwandte Methoden zum Einsatz, um eine gute Approximation bzgl. der H_∞-Norm zu erhalten. Alternativ kann die Hankel-Norm von (1) verwendet werden. Diese ist definiert als der maximale Hankel-Singulärwert des Systems. Formeln zur Berechnung der (für gegebenes $r < n$) optimalen Hankel-Norm-Approximation an ein stabiles System G findet man z.B. in [4, 29].

In der Literatur findet sich eine große Vielfalt verschiedener Techniken zur Modellreduktion, die. z.T. auch unterschiedlichen Zwecken dienen. Bei linearen Systemen scheinen modale Methoden (Craig-Bampton-Verfahren) und verwandte Techniken wie Substrukturierung und statische Kondensation, Padé- und Padé-artige Approximationsmethoden sowie Balancierungs-basierte Abschneidetechniken die wichtigste Rolle zu spielen; siehe die neueren Monographien und Übersichtsartikel [4, 5, 8, 10, 6, 18, 29, 30]. Die Modell- und Reglerreduktions-Funktionen in SLICOT basieren alle auf den letzten Ansatz. Einer der Gründe liegt darin, dass SLICOT keine speziellen Annahmen über die Besetzungsstruktur der Matrizen in (1) macht. Sowohl modale als auch Padé-Techniken haben höchstens dann Vorteile gegenüber dem Balancierten Abschneiden, wenn die Systemmatrizen dünnbesetzt sind und das System zu groß ist, um mit Methoden der numerischen linearen Algebra für dichtbesetzte Matrizen behandelt zu werden. Im Hinblick auf ihre Modellreduktionsfähigkeiten sind diese Methoden dem balancierten Abschneiden i.d.R. deutlich unterlegen, siehe z.B. [7] für einige Vergleiche.

Es gibt eine Reihe von Möglichkeiten, balanciertes Abschneiden zu interpretieren. Eine Möglichkeit liefert eine Analogie zur Best-Approximation einer Matrix, die durch die abgeschnittene Singulärwertzerlegung realisiert wird. Dabei bleiben die größten Singulärwerte der Matrix in der Approximation erhalten, während die kleinsten "abgeschnitten", also auf Null gesetzt,

werden. Die Rolle der Singulärwerte spielen hier nun die Hankel-Singulärwerte. Betrachte dazu die Steuerbarkeits-Gramsche W_c und die Beobachtbarkeits-Gramsche W_o des Systems (1). Diese sind im stabilen Fall gerade die eindeutigen Lösungen der Lyapunovgleichungen

$$\begin{aligned} AW_c + W_cA^T + BB^T &= 0, \\ A^TW_o + W_oA + C^TC &= 0. \end{aligned} \tag{12}$$

Ein LTI System ist *balanciert*, falls $W_c = W_o = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, wobei $\sigma_1 \geq \ldots \geq \sigma_n > 0$ gerade die Hankel-Singulärwerte des LTI Systems (1) sind. Die σ_i sind Systeminvarianten, sie bleiben bei Zustandsraumtransformationen unverändert. Das reduzierte Modell erhält man nun durch Beibehaltung der größten und Abschneiden der kleinsten Hankel-Singulärwerte, wobei die Realisierung $(\hat{A}, \hat{B}, \hat{C}, D)$ durch die führenden $r \times r$, $r \times m$, $p \times r$ und $p \times m$ Hauptabschnittsmatrizen der balancierten Realisierung (A, B, C, D) gegeben ist. Man beachte, daß es dazu nicht notwendig ist, diese Realsierung oder die balancierende Zustandsraumtransformation explizit zu berechnen. Gilt $\sigma_r > \sigma_{r+1}$ (r wird i.d.R. so gewählt), dann hat das reduzierte Modell einige wünschenswerte Eigenschaften: Stabilität bleibt erhalten und es existieren die berechenbaren Fehlerschranken

$$\sigma_{r+1} \leq \|G - \hat{G}\|_\infty \leq 2 \sum_{k=r+1}^{n} \sigma_k, \tag{13}$$

die eine adaptive Anpassung der Größe des reduzierten Modells an eine vorgegebene Fehlertoleranz ermöglichen. In der SLICOT Model and Controller Reduction Toolbox gibt es daneben noch eine Reihe weiterer balancierungsartiger Modellreduktionsmethoden, die z.T. auch andere Systemeigenschaften wie z.B. Minimalphasigkeit im reduzierten Modell erhalten können.

Ein Nachteil des balancierten Abschneidens ist, dass das statische Verhalten des Systems nicht erhalten bleibt, da man im allgemeinen $G(0) \neq \hat{G}(0)$ erwarten muss. Dies lässt sich durch Kombination mit statischer Kondensation (im Englischen auch "singular perturbation approximation") beheben. Auch balancierungsartige Methoden können mit statischer Kondensation kombiniert werden.

4.1 Funktionalität der Toolbox

Die SLICOT Model and Controller Reduction Toolbox setzt ausschliesslich auf theoretisch fundierte, numerisch verlässliche und effiziente Techniken, wie zum Beispiel balanciertes Abschneiden, statische Kondensation, balanciertes stochastisches Abschneiden, frequenzgewichtetes Balancieren, Hankel-Norm-Approximation, teilerfremde Faktorisierungen, usw. Eine detaillierte Übersicht der in SLICOT enthaltenen Fortran-Routinen zur Modell- und Reglerreduktion findet sich in [35, 37]. Die mathematischen Grundlagen der meisten dieser Techniken werden in [4, 29] dargestellt. Die Toolbox enthält:

- Ordnungsreduktion für kontinuierliche/diskrete LTI-Systeme und Regler;
- Ordnungsreduktion für stabile/instabile Systeme/Regler;
- Absolut- und Relativfehler-Methoden zur Modell- und Reglerreduktion;
- frequenzgewichtete Reduktion mit speziellen, Stabilität und Performanz erhaltenden Gewichtsfunktionen;
- Reduktion von zustands- und beobachterbasierten Reglern mittels teilerfremder Faktorisierungen.

Einige besondere Merkmale der Toolbox:

- numerische Zuverlässigkeit durch verschiedene, die Genauigkeit erhöhende Techniken (square-root, balancierungsfrei);
- hohe numerische Effizienz durch den Einsatz neuerer algorithmischer Entwicklungen und strukturausnutzender Verfahren;
- Matrixberechnungen basierend auf LAPACK und BLAS;
- Flexibilität und Benutzerfreundlichkeit;
- erweiterte Funktionalitäten, z.B. bei der Reglerreduktion;
- standardisierte Schnittstellen.

Tabelle 2 gibt eine Übersicht der in der Toolbox enthaltenen MATLAB-Funktionen für die Modell- und Reglerreduktion.

4.2 Performance-Resultate

Dieser Abschnitt enthält einige repräsentative experimentelle Ergebnisse für die in der SLICOT Model and Controller Reduction Toolbox enthaltenen Funktionen.[7]

Wir vergleichen zunächst die SLICOT-Funktion `bta` mit der entsprechenden Funktion `balred` aus der MATLAB Control System Toolbox sowie der Funktion `balancmr` aus der MATLAB Robust Control Toolbox. Zu diesem Zwecke werden Systemmatrizen A, B, C mit normalverteilten Pseudozufallszahlen erzeugt und A ersetzt durch $A \leftarrow A - \mu I$, so dass A asymptotisch stabil mit Stabilitätsreserve ca. 0.01. Abbildung 2 zeigt die resultierenden Laufzeiten der Modellreduktionsfunktionen für Systeme wachsender Ordnung im single-input/single-output (SISO, $m = p = 1$) und multi-input/multi-output (MIMO, $m = p = 10$ hier) Fall. In allen Fällen wird ein reduziertes Modell der Ordnung $r = 20$ berechnet. Augenscheinlich ist `bta` immer die schnellste Funktion; im Vergleich zu der bereits recht effizienten MATLAB-Funktion `balred` wird rund 40% der Laufzeit eingespart. Dieser Geschwindigkeitsvorteil wird noch deutlicher im Vergleich zu früheren MATLAB-Implementierungen wie `balancmr`. Die Genauigkeit der reduzierten Modelle ist bei allen Funktionen vergleichbar.

[7] Alle Tests wurden auf einem Samsung M50 mit 1 GB Hauptspeicher, Intel M 2.13 Ghz Prozessor in MATLAB R2006a durchgeführt.

Tabelle 2: MATLAB-Funktionen in der SLICOT Model and Controller Reduction Toolbox.

Funktion	Beschreibung
`bta`	Modellreduktion mittels balanciertem Abschneiden des stabilen Anteils (square-root, balancierungsfrei)
`btabal`	Modellreduktion mittels balanciertem Abschneiden des stabilen Anteils (square-root)
`bta_cf`	Modellreduktion mittels balanciertem Abschneiden der teilerfremden Faktoren (square-root, balancierungsfrei)
`btabal_cf`	Modellreduktion mittels balanciertem Abschneiden der teilerfremden Faktoren (square-root)
`spa`	Modellreduktion mittels statischer Kondensation des stabilen Anteils (square-root, balancierungsfrei)
`spabal`	Modellreduktion mittels statischer Kondensation des stabilen Anteils (square-root)
`spa_cf`	Modellreduktion mittels statischer Kondensation der teilerfremden Faktoren (square-root, balancierungsfrei)
`spabal_cf`	Modellreduktion mittels statischer Kondensation der teilerfremden Faktoren (square-root)
`hna`	Modellreduktion mittels Hankel-Norm-Approximation des stabilen Anteils (square-root)
`bst`	Modellreduktion mittels balancierten stochastischen Abschneidens
`fwbred`	Modellreduktion mittels frequenzgewichteten Balancierens
`fwhna`	Modellreduktion mittels frequenzgewichteter Hankel-Norm-Approximation
`fwbconred`	Reglerreduktion mittels frequenzgewichteten Balancierens
`sfconred`	Zustandsbasierte Reglerreduktion mittels teilerfremder Faktorisierung

Exemplarisch betrachten wir noch ein weiteres Beispiel und vergleichen die Funktionen zum balancierten stochastischen Abschneiden `bst` in SLICOT und `bstmr` in der MATLAB Robust Control Toolbox.[8] Im Vergleich zum balancierten Abschneiden wird hier die Beobachtbarkeits-Gramsche durch die Lösung einer algebraischen Riccati-Gleichung ersetzt. Balanciertes stochastisches Abschneiden gehört zur Klasse der Relativfehler-Methoden zur Modellreduktion und erhält sowohl Stabilität als auch Minimalphasigkeit. Der obere Graph in Abbildung 3 zeigt die gemessenen Laufzeiten beider Funktionen für stabile SISO Systeme wachsender Ordnung. Wiederum ist die SLICOT-Funktion erheblich schneller als die entsprechende Funktion der MATLAB-Toolbox.

Der untere Graph in Abbildung 3 soll einen Eindruck vermitteln, welche Genauigkeit mit balancierten stochastischen Abschneiden erzielt werden kann. Hier verwenden wir ein populäres Beispiel aus der SLICOT Benchmarksammlung zur Modellreduktion [16]: ISS. Dieser Datensatz hat die Dimensionen $n = 270$, $m = p = 3$ und beschreibt eine Komponente der internationalen Raumstation. Gezeigt wird ein Teil des Bode-Diagramms für

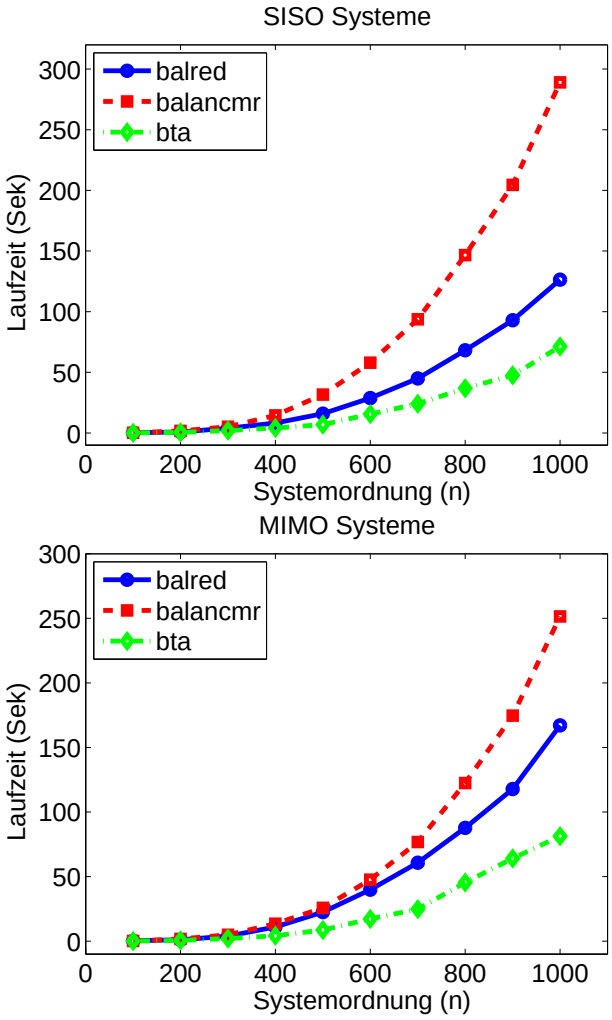


Bild 2: Vergleich der verschiedenen MATLAB-Funktionen für balanciertes Abschneiden von SISO (oben) und MIMO (unten, $m = p = 10$) Systemen.

den Fehler $G(s) - \hat{G}(s)$: der Absolutbetrag des Fehlers der Übertragungsfunktion vom dritten Eingang zum ersten Ausgang.[9] Hier schneidet `bst` ein wenig besser für niedrigere Frequenzen ab.

Es soll aber betont werden, dass – trotz der in diesem Abschnitt demonstrierten deutlichen Geschwindigkeitsvorteile – die Hauptstärke der SLICOT Model and Controller Reduction Toolbox die Verfügbarkeit von frequenzgewichteten Varianten des balancierten Abschneidens zur Modell- und Reglerreduktion ist. Diese sind momentan in keiner MATLAB-Toolbox enthalten. Die SLICOT-Toolbox stellt also eine wesentlich reicheres Arsenal von Modellreduktionstechniken zur Verfügung.

Literatur

[1] J. Abels and P. Benner. CAREX - a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). SLICOT working note 1999-14, 1999. Available online from `http://www.slicot.org`.

[8] Balanciertes stochastisches Abschneiden ist in der MATLAB Control System Toolbox nicht verfügbar.

[9] Die Resultate unterscheiden sich nur unwesentlich für die anderen Eingangs-Ausgangs-Kanäle.

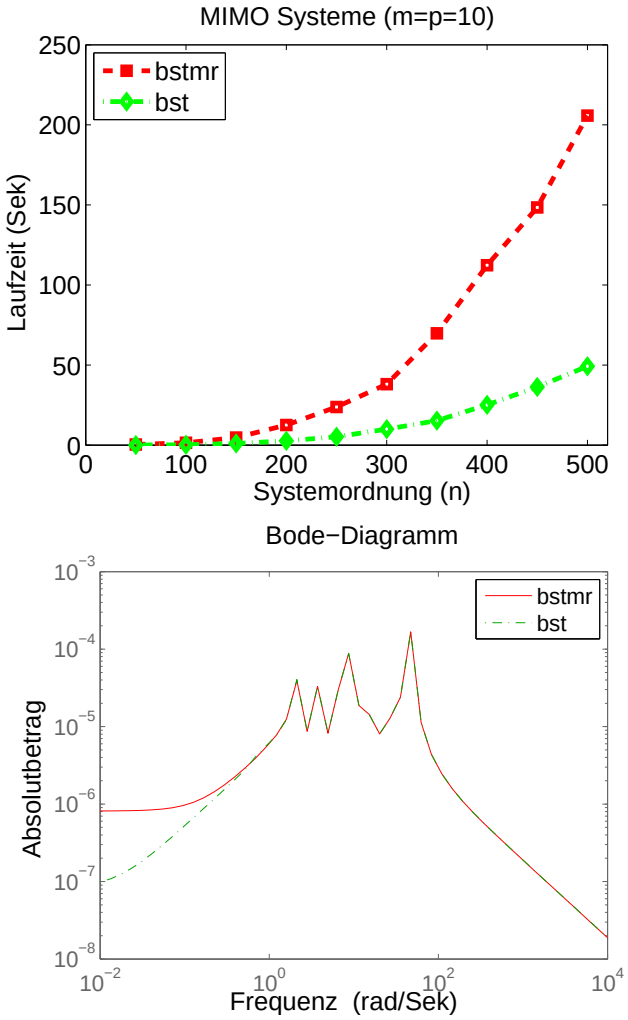


Bild 3: Vergleich von MATLAB-Funktionen zum balancierten stochastischen Abschneiden: Laufzeiten (oben) und Genauigkeit (unten).

[2] J. Abels and P. Benner. DAREX - a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0). SLICOT working note 1999-16, 1999. Available online from `http://www.slicot.org`.

[3] A. Angermann, M. Beuschel, M. Rau, and U. Wohlfarth. *MATLAB – Simulink – Stateflow: Grundlagen, Toolboxen, Beispiele*. Oldenbourg, 5 edition, 2007.

[4] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.

[5] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. In *Structured matrices in mathematics, computer science, and engineering, I (Boulder, CO, 1999)*, volume 280 of *Contemp. Math.*, pages 193–219. Amer. Math. Soc., Providence, RI, 2001.

[6] Z. Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Appl. Numer. Math.*, 43(1-2):9–44, 2002.

[7] P. Benner. Numerical linear algebra for model reduction in control and simulation. *GAMM Mitteilungen*, 29:275–296, 2006.

[8] P. Benner, R. W. Freund, D. C. Sorensen, and A. Varga, editors. *Special Issue on Order Reduction of Large-Scale Systems*, volume 415, issues 2–3 of *Linear Algebra and Its Applications*. June 2006.

[9] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT—a subroutine library in systems and control theory. In *Applied and computational control, signals, and circuits, Vol. 1*, pages 499–539. Birkhäuser Boston, Boston, MA, 1999. See also `http://www.slicot.de`.

[10] P. Benner, V. Mehrmann, and D. C. Sorensen, editors. *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[11] P. Benner and V. Sima. Solving algebraic Riccati equations with SLICOT. In *Proceedings of the 11th Mediterranean Conference on Control & Automation MED'03, June 18-20, 2003, Rhodes*, 2003.

[12] P. Benner, V. Sima, and M. Slowik. Evaluation of the linear matrix equation solvers in SLICOT. *JNAIAM J. Numer. Anal. Ind. Appl. Math.*, 2(1-2):11–34, 2007.

[13] S. Boyd and V. Balakrishnan. A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its $\mathbf{L}_\infty$-norm. *Systems Control Lett.*, 15(1):1–7, 1990.

[14] R. N. Bracewell. *The Hartley transform*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1986. Oxford Engineering Science Series, 19.

[15] N. A. Bruinsma and M. Steinbuch. A fast algorithm to compute the H_∞-norm of a transfer function matrix. *Sys. Control Lett.*, 14(4):287–293, 1990.

[16] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. SLICOT working note 2002-2, 2002. Available online from `http://www.slicot.org`.

[17] B. N. Datta. *Numerical Methods for Linear Control Systems Design and Analysis*. Elsevier Academic Press, 2003.

[18] R. W. Freund. Model reduction methods based on Krylov subspaces. *Acta Numer.*, 12:267–319, 2003.

[19] M. Green and D. J. N. Limebeer. *Linear Robust Control*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[20] S. Hammarling. Numerical solution of the stable, nonnegative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2(3):303–323, 1982.

[21] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, second edition, 2002.

[22] T. Kailath and A.H. Sayed. Displacement structure: theory and applications. *SIAM Review*, 37:297–386, 1995.

[23] D. Kressner and P. Van Dooren. Factorizations and linear system solvers for matrices with Toeplitz structure. SLICOT working note 2000-2, June 2000. Updated June 2001. Available online from `http://www.slicot.org`.

[24] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation*. Oxford University Press, Oxford, 1995.

[25] A. J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control*, AC-24:913–921, 1979.

[26] The MathWorks, Inc., Natick, Mass, 01760. *The* MATLAB *Control Toolbox, Version 8.2*, 2008.

[27] V. Mehrmann and T. Stykel. Descriptor systems: A general mathematical framework for modelling, simulation and control. *at-Automatisierungstechnik*, 54(8):405–415, 2006.

[28] P. Misra, P. Van Dooren, and A. Varga. Computation of structural invariants of generalized state-space systems. *Automatica*, 30(12):1921–1936, 1994.

[29] G. Obinata and B. D. O. Anderson. *Model Reduction for Control System Design*. Communications and Control Engineering Series. Springer-Verlag, London, UK, 2001.

[30] W. H. A. Schilders, H. A. van der Vorst, and J. Rom-

mes, editors. *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Series Mathematics in Industry*. Springer-Verlag, 2008.

[31] V. Sima. Performance investigation of SLICOT Wiener systems identification toolbox. In P. Van den Hof, B. Wahlberg, and S. Weiland, editors, *Proceedings of The 13th IFAC Symposium on System Identification, SYSID 2003, August 27–29, 2003, Rotterdam, The Netherlands*, pages 1345–1350. Omnipress, 2003.

[32] V. Sima. Computational experience with subspace identification tools. In *12th Mediterranean Conference on Control and Automation MED'04, June 6–9 2004, Kusadasi, Aydin / Turkey*, 2004.

[33] V. Sima, D. M. Sima, and S. Van Huffel. High-performance numerical algorithms and software for subspace-based linear multivariable system identification. *J. Comput. Appl. Math.*, 170(2):371–397, 2004.

[34] P. Van Overschee and B. De Moor. N4SID: subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica J. IFAC*, 30(1):75–93, 1994.

[35] A. Varga. Controller reduction using accuracy-enhancing methods. Chapter 9 (pages 225–260) of [10].

[36] A. Varga. A descriptor systems toolbox for MATLAB. In *Proc. of IEEE International Symposium on Computer Aided Control System Design, CACSD'2000, Anchorage, Alaska*, pages 150–155, 2000.

[37] A. Varga. Model reduction software in the SLICOT library. In B.N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 629 of *The Kluwer International Series in Engineering and Computer Science*, pages 239–282. Kluwer Academic Publishers, Boston, MA, 2001.

[38] A. Varga. Numerical software in SLICOT for low order controller design. In *Proc. of CACSD'2002, Glasgow, UK*, 2002.

[39] M. Verhaegen and P. Dewilde. Subspace model identification. I. The output-error state-space model identification class of algorithms. *Internat. J. Control*, 56(5):1187–1210, 1992.

[40] D. Westwick and M. Verhaegen. Identifying MIMO Wiener systems using subspace model identification methods. *Signal Processing*, 52:235–258, 1996.

Manuskripteingang: .

Prof. Dr. rer. nat. Peter Benner ist Inhaber der Professur Mathematik in Industrie und Technik in der Fakultät für Mathematik der Technischen Universität Chemnitz. Hauptarbeitsgebiete: Numerische Methoden für Modellreduktion, Regelung und optimale Steuerung dynamischer Systeme sowie Numerische Lineare Algebra und Parallele Algorithmen.

Adresse: Technische Universität Chemnitz, Fakultät für Mathematik, D-09107 Chemnitz, Deutschland, Fax: +49-(0)371-531-22509, E-Mail: benner@mathematik.tu-chemnitz.de

Prof. Dr. rer. nat. Daniel Kressner ist Assistenzprofessor im Fachgebiet Numerische Mathematik am Departement Mathematik der ETH Zürich. Hauptarbeitsgebiete: Numerische lineare Algebra, insbesondere Eigenwertprobleme, sowie die Entwicklung numerischer Verfahren in der System- und Regelungstheorie.

Adresse: ETH Zürich, Rämistr. 101, 8092 Zürich, Schweiz, E-Mail: kressner@math.ethz.ch

Dr. Ing. Math. Vasile Sima ist Senior Researcher ersten Grades am National Institute for Research & Development in Informatics (ICI), Bucharest. Hauptarbeitsgebiete: Adaptive and optimale Steuerung, computergestützte Regelungs- und Systemtheorie, numerische lineare Algebra und wissenschaftliches Rechnen, nichtlineare Optimierung, sowie Systemidentifizierung.

Adresse: ICI, Bd. Maresal Averescu, Nr. 8-10, 011455, Bukarest, Rumänien, E-Mail: vsima@ici.ro

Dr. Ing. Andras Varga ist Senior Scientist am Deutschen Zentrum für Luft- und Raumfahrt (DLR), Institut für Robotik und Mechatronik in Oberpfaffenhofen. Hauptarbeitsgebiete: Entwicklung numerischer Verfahren und Software für Regelungs- und Systemtheorie mit Schwerpunkten Modell- und Reglerordnungsreduktion, Deskriptorsysteme, periodische Systeme und Fehlerdetektion.

Adresse: DLR Oberpfaffenhofen, Institut für Robotik und Mechatronik, 82234 Wessling, Deutschland. E-Mail: andras.varga@dlr.de